IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BOARD OF PATENT APPEALS AND INTERFERENCES

In re patent application of:
Glider et al.

Serial No.: 10/723,085

Filed: November 26, 2003

Group Art Unit: 2192

Examiner: Zheng Wei

Atty. Docket No.: ARC920030081US1

For: SOFTWARE UPGRADE AND DOWNGRADE IN SYSTEMS WITH PERSISTENT DATA

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' APPEAL BRIEF

Sirs:

Appellants respectfully appeal the final rejection of claims 1-5, 7-11, 13, and 15-19, in the Office Action dated March 4, 2008. A Notice of Appeal and Pre-Appeal Brief Request for Review was timely filed on June 2, 2008.

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, Armonk, New York, assignee of 100% interest of the above-referenced patent application.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-5, 7-11, 13, and 15-19 are all the claims pending in the application and are set forth fully in the attached appendix (Section IX), are under appeal. Claims 1-20 were originally filed in the application. Claims 6, 12, 14 and 20 are cancelled. A non-final Office Action was issued on August 21, 2007 rejecting claims 1-5, 7-11, 13, and 15-19. The Appellants filed an Amendent under 37 C.F.R. §1.111 on November 20, 2007 amending claims 1-5, 7-11, 13, and 15-19. A final Office Action was issued on March 4, 2008 rejecting claims 1-5, 7-11, 13, and 15-19. The Appellants filed an Amendent under 37 C.F.R. §1.116 on May 5, 2008 amending dependent claims 2-5, 7-11, 13, and 15-19. An Advisory Action was issued on June 13, 2008, indicating that the Amendent filed under 37 C.F.R. §1.116 on May 5, 2008 would be entered for purposes of appeal and an explanation of how the amended claims would be rejected was appended. The Appellants had already filed a timely Notice of Appeal on June 2, 2008.

Claims 1-5, 7-11, 13, and 15-19 stand rejected under 35 U.S.C. §103(a) as unpatentable over U.S. Patent Application Publication No. 2003/0092438 to Moore et al., hereinafter, Moore, in view of U.S. Patent No. 6,385,770 to Sinander et al., hereinafter,

Sinander, and further in view of U.S. Patent No. 7,107,329 to Schroder et al., hereinafter, Schroder.

Appellants respectfully traverse the rejection based on the following discussion.

IV. STATEMENT OF AMENDMENTS

A final Office Action dated March 4, 2008 stated all the pending claims 1-5, 7-11, 13, and 15-19 were rejected. The claims shown in the appendix (Section IX) are shown in their amended form as of the May 5, 2008 amendment.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The Summary of the claimed subject matter is shown by the parenthetically, annotated, independent claims, designating page and line numbers of the Specification, and Figures are followed by an underlined element number, where appropriate:

Independent Claim 1

A method for revising a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said method comprising:

applying an upgrade to a first next level of software that understands both old and new persistent data structure formats (p. 8, l. 22 to p. 9, l. 2; Fig. 4, 100);

converting all persistent data structures into the old persistent data structure format (p. 9, l. 2-3; Fig. 4, 110);

applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats (p. 9, l. 3-4; Fig. 4, 120);

converting all persistent data structures into the new persistent data structure format (p. 9, l. 4-5; Fig. 4, 130);

applying a downgrade to a first previous level of software that understands both

said old and new persistent data structure formats (p. 9, l. 6-7; Fig. 5, 140);

converting all persistent data structures into the old persistent data structure format (p. 9, l. 7-8; Fig. 5, 150); and

applying a downgrade to a second previous level of software that understands said old persistent data structure formats (p. 9., l. 8-9; Fig. 5, 160),

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network (p. 9, l. 19-22), and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes (p. 9, l. 19-22).

Independent Claim 7

A system for providing updates to a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said system comprising:

a first module operable for applying an upgrade to a first next level of software that understands both old and new persistent data structure formats (p. 15, l. 3-5; Fig. 7, 710);

a first converter in said first module operable for converting all persistent data structures into the old persistent data structure format (p. 15, l. 5-6; Fig. 7, 715);

a second module operable for applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats (p. 15, l. 6-8; Fig. 7, 720);

a second converter in said second module operable for converting all persistent data structures into the new persistent data structure format (p. 15, l. l8-9; Fig. 725)

a third module operable for applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats (p. 15, l. 9-11; Fig. 7, 730);

a third converter in said third module operable for converting all persistent data

structures into the old persistent data structure format (p. 15, l. 11-13; Fig. 7, 735); and

a fourth module operable for applying a downgrade to a second previous level of software that understands said old persistent data structure formats (p. 15, l. 13-14; Fig. 7, 740),

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network (p. 9, l. 19-22), and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes (p. 9, l. 19-22).

Independent Claim 13

A system for providing updates to a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said system comprising:

means for applying an upgrade to a first next level of software that understands both old and new persistent data structure formats (p. 14, l. 14-15);

means for converting all persistent data structures into the old persistent data structure format (p. 14, l. 15-16);

means for applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats (p. 14, l. 16-17);

means for converting all persistent data structures into the new persistent data structure format (p. 14, l. 18)

means for applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats (p. 14, l. 19-20);

means for converting all persistent data structures into the old persistent data structure format (p. 14, l. 20-21); and

means for applying a downgrade to a second previous level of software that understands said old persistent data structure formats (p. 14, l. 21-22),

wherein the nodes are adapted to communicate with one another at a time when

said nodes are operating at different software levels with respect to one another within said computer network (p. 9, l. 19-22), and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes (p. 9, l. 19-22).

Independent Claim 15

A program storage device readable by computer, tangibly embodying a program of instructions executable by said computer to perform a method for revising a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said method comprising:

applying an upgrade to a first next level of software that understands both old and new persistent data structure formats (p. 8, l. 22 to p. 9, l. 2; Fig. 4, 100);

converting all persistent data structures into the old persistent data structure format (p. 9, l. 2-3; Fig. 4, 110);

applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats (p. 9, l. 3-4; Fig. 4, 120);

converting all persistent data structures into the new persistent data structure format (p. 9, l. 4-5; Fig. 4, 130);

applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats (p. 9, l. 6-7; Fig. 5, 140);

converting all persistent data structures into the old persistent data structure format (p. 9, l. 7-8; Fig. 5, 150); and

applying a downgrade to a second previous level of software that understands said old persistent data structure formats (p. 9., l. 8-9; Fig. 5, 160),

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network (p. 9, l. 19-22), and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes (p. 9, l. 19-22).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented for review by the Board of Patents Appeals and Interferences are whether independent claims 1, 7, 13, and 15, and dependent claims 2-5, 8-11, and 16-19 are unpatentable under 35 U.S.C. §103(a) over Moore, Sinander and Schroder.

VII. ARGUMENT

A. The Prior Art Rejections Based of Claims 1-5, 7-11, 13, and 15-19

1. The Position in the Office Action

Claims 1, 7, 13 and 15:

Moore discloses a method and apparatus for revising a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said method comprising:

• Applying an upgrade to a next level of software (see for example, Fig. 4, step 118-120, UPGRADE and related text)

• Converting all persistent data structures to new version format (see for example, Fig. 4, step 120 CONVERT STAE DATA TO NEW VERSION FORMAT and related text)

• Applying a downgrade to a previous level of software. (see for example, Fig.3, items 102 and related text)

• Converting all persistent data structures into the old persistent data structure format. (see for example Fig.3, item 112 and related text)

• Applying a downgrade to a second previous level of software that understands said old persistent data structure formats. (Fig.4, items 116-122).

But does not explicitly disclose about two-level software upgrading. However, Sinander in the same analogous art of software upgrade discloses a method and system

for upgrading a software application utilizes all kinds of data, said method and system comprising:

• Applying an upgrade to a first part of an upgrade framework to upgrade system software; (Col 3, Lines 54-58)

• Executing a plurality of upgrade contents to convert data structure; (CoI2, Lines 6-16)

• Applying an upgrade to a second part of the upgrade frame to upgrade system software; (Col 3, Lines 54-58).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use Sinander's upgrade method combine with Moore's software upgrade/downgrade method. One would have been motivated to integrate Sinander's upgrade method to Moore's upgrade method as suggested by Sinander (see for example, ABSTRACT, "The invention allows to upgrade a software system in a real-time environment using a source system operating with an old software version and a target system for operating with the new software version and allows to handle static as well as dynamic data").

But neither of them further discloses both upgrade processes and both downgrade processed occur without disruption of communication between said nodes. However, Schroder in the same analogous art of upgrading software of network nodes discloses a method for updating routers (nodes) software in network without traffic interruption (see for example, Fig.1B, the upgrade process by using "hot swap" implementation, "Before Upgrade", During Upgrade", "After Upgrade" and related text).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use Moore and Sinander's method to prepare new software information including revisions and upgrades as address above and further use Schroder's method to swap the original and upgraded software in the node without service disruption. One would have been motivated to do so to support network node software/firmware upgrade without traffic interruption as suggested by Schroder (see for example, col.2, lines 18-24, "after such preparing of the new software information,

swapping the same for the original software data routing along said path without interruption, and imperceptibly to all the other router nodes in the router system")

Claim 2, 8 and 16:

Sinander, Moore and Schroder disclose a system and method to upgrade software application utilizes persistent data as in claims 1, 7, and 15 above, but does not explicitly disclose that the persistent data structures comprise communication packet structures. However, Sinander further discloses the system and method for software upgrade could be used in a real time application of telecommunications network (Co1. 1, Line 41-44) and switch communication links (CoI2, Line36). That would have been obvious to one having ordinary skill in the art at time the invention was made to understand that these networks, like ATM, IP networks use packet (ATM cells or IP packet) for communication based on different kinds of network protocols. Therefore, one would have been motivated to use persistent data structure to represent the packet structure in software programming in order to make software implementation simpler and easier.

Claim 3, 9 and 17:

Sinander, Moore and Schroder disclose a system and method to upgrade software application as in claims 2, 8 and 16 above and Sinander further discloses that the distributed system including a plurality of nodes (Co.10, lines 47-50, "In case the source system is operating a mobile telephone network, the devices may be mobile telephones or nodes of the network.") holding non-volatile memory data structure. (Co1.6, lines 36-48),

Claims 4, 10 and 18:

Sinander, Moore and Schroder disclose a system and method to upgrade software application as in claims 3, 9 and 17 above and Sinander also discloses that said nodes communicate with one another. (Co1.10, lines 47-50, "In case the source system is operating a mobile telephone network, the devices may be mobile telephones or nodes of the network."). Therefore, it is obvious for a person with ordinary skill in the art at time

the invention was made to understand that the "mobile telephone or nodes of the network" can communicate to each other.

Claims 5, 11 and 19:

Sinander, Moore and Schroder disclose a system and method to upgrade software application as in claims 4, 10 and 18 above and Sinander further discloses that said nodes communicate with one another. (Col. 10, lines 47-50, "In case the source system is operating a mobile telephone network, the devices may be mobile telephones or nodes of the network"). Therefore, it would have been obvious to one having ordinary skill in the art at time the invention was made to understand that said nodes, like mobile telephones or nodes in networks can use communication packet to communicate between each other.

In view the foregoing, the Board is respectfully requested to reconsider and withdraw these rejections.

2. The Prior Art References (Moore, Sinander and Schroder)

a. The Moore Disclosure

As shown in Figs 3 and 4, Moore discloses that an upgrade or downgrade service generally entails the installation of new application software or hardware on a secondary controller 54. At step 102, the secondary controller 54 has its application software or hardware 64 upgraded, or downgraded. At step 104, the secondary controller 54 prepares to assume control of the system 50; specifically, the secondary application 64 notifies the secondary control block version table 76 of the new application version number. Then, at step 106, the checkpoint service 82 communicates with the primary control block 70 and updates the control block version table 74 to indicate the new secondary application version. At step 107, the primary controller 52 begins to shutdown or quiesce. (Paragraph [0022], which is cited by the Office Action).

Moore also discloses that at step 108 the primary processor 58 reads the primary control block version table 74 and compares versions of the primary application 62 and secondary application 64. With the results, step 110 determines whether the change in

the secondary application was an upgrade, a downgrade or no change. If step 110 determines that the new version is a downgrade, step 112 converts the saved state data to a format compatible to the older version running on the secondary processor 60, rewrites the converted data to the replica state databases 78, notifies the checkpointing service 82 and updates the replica state database 80. The process then continues to step 113 as shown in Fig. 4. If step 110 determines that the new version is an upgrade or no change, conversions of the state data may be delayed, and the process continues with step 114 wherein the primary controller 52 performs a shutdown of services and passes control of the system 50 to the secondary controller 54. (Paragraph [0023, which is cited by the Office Action).

Moore further discloses that at step 116, the secondary controller 54 takes over primary control of the system 50 and opens the checkpoint replica database 80 for read and write access. Next, at step 118, the system determines if the replica state data needs to be converted (i.e., the new application is an upgrade). If the replica state needs to be converted, step 120 converts the data to the new version format. After conversion, the secondary processor 60, which now controls the system 50, imports state data fro the replica state database 80 to the local database at step 122. At step 122, normal operations may resume. (Paragraph [0024], which is cited by the Office Action).

b. The Sinander Disclosure

Sinander discloses combining identical upgrade tasks of individual upgrades and executing a plurality of sequential content upgrades, each corresponding to a single upgrade, in an upgrade framework (col. 2, lines 6-10, which are cited by the Final Action). The upgrade framework may also comprise tasks for forwarding dynamic data between dynamic tasks of sequential upgrade contents (col. 2, lines 57-59). The upgrade framework may at least comprise dumping databases or contents of data storage means from the source system to the target system and initializing the logging of dynamic data in an event log (col.2, lines 63-66).

Sinander also discloses that the upgrade operation is partitioned into an upgrade

framework, the upgrade framework comprising tasks identical for each of the plurality software system upgrades, and into upgrade contents, each of the upgrade contents comprising tasks specific to the corresponding software system upgrade (col. 3, line 40-45).

Sinader further discloses that a first part of the upgrade framework consists of upgrade tasks to be executed at an initial stage of the upgrade operation, before executing the upgrade contents. A second part of the upgrade framework consists of upgrade tasks to be executed in a final stage of the upgrade operation (col. 3, lines 54-58, which are cited by the Office Action).

c. The Schroder Disclosure

Schroder discloses preparing new software information at one node from the original software and including revisions and upgrades; and after such preparing of the new software information in the one node during the continuing of the data routing along the path without interruption (col. 2, lines 16-21, which are cited by the Office Action).

3. The Appellants' Position regarding Independent Claims 1, 7, 13, and 15, and Dependent Claims 2-5, 8-11, and 16-19

Independent claims 1, 7, 13, and 15 recite in relevant part,

"applying an upgrade to a first next level of software … ;

…

applying an upgrade to a second next level of software … ;

…

applying a downgrade to a first previous level of software … ;

…

applying a downgrade to a second previous level of software … ."

With regard to the cited reference of Moore, the final Office Action admits, "But

[Moore] does not explicitly disclose about two-level software upgrading." (Please see, final Office Action, page 5, printed line 5).

The final Office Action cites Sinander for disclosing a first part of an upgrade framework and a second part of an upgrade framework, which are asserted to be analogous to the applying an upgrade to a first next level of software and to the applying an upgrade to a second next level of software (and similarly, to the applying a downgrade to a first previous level of software and to the applying a downgrade to a second previous level of software) of the present invention. (Please see, Final Action, page 5, second paragraph, bullet points, and third paragraph). (emphases added).

However, Appellants respectfully submit that the first and second parts of the "upgrade framework" of Sinander are not analogous to the first and second next/previous levels of software to which upgrades/downgrades are applied. Instead, the first part of the upgrade framework of Sinander is defined as "consist[ing] of upgrade tasks to be executed at an initial stage of the upgrade operation, before executing the upgrade contents', while the second part of the "upgrade framework" of Sinander is defined as "consist[ing] of upgrade tasks to be executed in a final stage of the upgrade operation" (Sinander, (col. 3, lines 54-58). That is, the first and second parts of the upgrade framework of Sinander are, respectively, (1) identical (and common) tasks (or processes) that are to be performed initially in an upgrade to a version of software, and (2) after performing the initial tasks (i.e., first part of upgrade framework) then performing the upgrading of the contents of the upgrade (i.e., the second part of the upgrade framework).

In contrast, the present invention describes applying an upgrade (downgrade) to a first next (previous) level of software and applying an upgrade (downgrade) to a second next (previous) level of software. Upgrades and downgrades, not first and second groups of processes involved in an upgrade as described by Sinander, are applied to first and second levels of software (i.e., that is, a level of a software application in its entirety).

For at least the reasons outline above with respect to the disclosure of Sinander, Appellants respectfully submit that Sinander does not disclose, teach or suggest at least

the present invention's features of "applying an upgrade to a first next level of software … ; … applying an upgrade to a second next level of software … ; … applying a downgrade to a first previous level of software … ; … applying a downgrade to a second previous level of software … .", as recited in independent claims 1, 7, 13, and 15.

In networks of interconnected router nodes for forwarding data traffic along a predetermined path, Schroder merely discloses a method and system for imperceptibly upgrading router node software without traffic interruption through preparation of upgraded software in a router, while the router continues to forward data under the control of its original software, and then swapping the upgraded software for the original software without disruption. (Abstract, lines 1-8).

Appellants respectfully submit that Schroder does not cure the deficiencies of Moore and Sinander.

Nowhere does Schroder disclose, teach or suggest the present invention's feature of "applying an upgrade to a first next level of software … ; … applying an upgrade to a second next level of software … ; … applying a downgrade to a first previous level of software … ; … applying a downgrade to a second previous level of software … .", as recited in independent claims 1, 7, 13, and 15

For at least the reasons outlined immediately above, Appellants respectfully submit that Moore, Sinander, and Schroder, either individually or in combination, do not disclose, teach or suggest the present invention's features of: "applying an upgrade to a first next level of software … ; … applying an upgrade to a second next level of software … ; … applying a downgrade to a first previous level of software … ; … applying a downgrade to a second previous level of software … ", as recited in independent claims 1, 7, 13, and 15. Accordingly, Moore, Sinander, and Schroder, either individually or in combination, fail to render obvious the subject matter of independent claims 1, 7, 13, and 15, and dependent claims 2-5, 8-11, and 16-19 under 35 U.S.C. §103(a).

In view of the foregoing, the Board is respectfully requested to reconsider and withdraw the rejection of claims 1-5, 7-11, 13, and 15-19 under 35 U.S.C. §103(a) as unpatentable over Moore, Sinander, and Schroder.

VIII. CONCLUSION

In view of the foregoing, the Appellants respectfully submit that the cited prior art does not disclose, teach or suggest the present invention's features described by independent claims 1, 7, 13, and 15, and as such, claims 1, 7, 13 and 15 are patentable over Moore, Sinander and Schroder, either individually or in combination. Furthermore, Appellants respectfully submit that for the identical reasons outlined above with respect to the rejection of the independent claims over Moore, Sinander, and Schroder, the dependent claims 2-5, 8-11 and 16-19 are also patentable over Moore, Sinander and Schroder, either individually or in combination, by virtue of their dependency from the patentable independent claims.

Therefore, Appellants respectfully request that the Board reconsider and withdraw the rejections of all of the pending claims, i.e., claims 1-5, 7-11, 13, and 15-19 and pass these claims to issue.

Please charge any deficiencies and credit any overpayments to Attorney's Deposit Account Number 09-0441.

Respectfully submitted,

Date: August 4, 2008

/Peter A. Balnave/
Peter A. Balnave, Ph.D.
Registration No. 46,199

Gibb & Rahman, LLC
2568-A Riva Road, Suite 304
Annapolis, MD, 21401
Voice: (410) 573-5255
Fax: (301) 261-8825
Customer No. 29154

IX. CLAIMS APPENDIX

1. A method for revising a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said method comprising:

applying an upgrade to a first next level of software that understands both old and new persistent data structure formats;

converting all persistent data structures into the old persistent data structure format;

applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats;

converting all persistent data structures into the new persistent data structure format;

applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats;

converting all persistent data structures into the old persistent data structure format; and

applying a downgrade to a second previous level of software that understands said old persistent data structure formats,

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network, and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes.

2. The method according to claim 1, wherein said persistent data structures comprise communication packet structures.

3. The method according to claim 2, wherein said software application comprises a distributed system software application, and wherein said plurality of nodes hold non-volatile memory data structures.

4. The method according to claim 3, wherein said nodes communicate with one another.

5. The method according to claim 4, wherein the communication between said nodes occurs using said communication packet structures.

6. (Cancelled).

7. (Previously Presented) A system for providing updates to a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said system comprising:

a first module operable for applying an upgrade to a first next level of software that understands both old and new persistent data structure formats;

a first converter in said first module operable for converting all persistent data structures into the old persistent data structure format;

a second module operable for applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats;

a second converter in said second module operable for converting all persistent data structures into the new persistent data structure format

a third module operable for applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats;

a third converter in said third module operable for converting all persistent data structures into the old persistent data structure format; and

a fourth module operable for applying a downgrade to a second previous level of software that understands said old persistent data structure formats,

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network, and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes.

8. The system according to claim 7, wherein said persistent data structures comprise communication packet structures.

9. The system according to claim 8, wherein said software application comprises a distributed system software application, and wherein said plurality of nodes hold non-volatile memory data structures.

10. The system according to claim 9, wherein said nodes communicate with one another.

11. The system according to claim 10, wherein the communication between said nodes occurs using said communication packet structures.

12. (Cancelled).

13. (Previously Presented) A system for providing updates to a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said system comprising:

means for applying an upgrade to a first next level of software that understands both old and new persistent data structure formats;

means for converting all persistent data structures into the old persistent data structure format;

means for applying an upgrade to a second next level of software that understands

both said old and new persistent data structure formats;

means for converting all persistent data structures into the new persistent data structure format

means for applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats;

means for converting all persistent data structures into the old persistent data structure format; and

means for applying a downgrade to a second previous level of software that understands said old persistent data structure formats,

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network, and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes.

14. (Cancelled).

15. (Previously Presented) A program storage device readable by computer, tangibly embodying a program of instructions executable by said computer to perform a method for revising a software application used by a plurality of nodes in a computer network, wherein said software application utilizes persistent data, said method comprising:

applying an upgrade to a first next level of software that understands both old and new persistent data structure formats;

converting all persistent data structures into the old persistent data structure format;

applying an upgrade to a second next level of software that understands both said old and new persistent data structure formats;

converting all persistent data structures into the new persistent data structure format

applying a downgrade to a first previous level of software that understands both said old and new persistent data structure formats;

converting all persistent data structures into the old persistent data structure format; and

applying a downgrade to a second previous level of software that understands said old persistent data structure formats,

wherein the nodes are adapted to communicate with one another at a time when said nodes are operating at different software levels with respect to one another within said computer network, and

wherein both upgrade processes and both downgrade processes occur without disruption of communication between said nodes.

16. The program storage device according to claim 15, wherein said persistent data structures comprise communication packet structures.

17. The program storage device according to claim 16, wherein said software application comprises a distributed system software application, and wherein said plurality of nodes hold non-volatile memory data structures.

18. The program storage device according to claim 17, wherein said nodes communicate with one another.

19. The program storage device according to claim 18, wherein the communication between said nodes occurs using said communication packet structures.

20. (Cancelled).

X. EVIDENCE APPENDIX

There is no other evidence known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There is no other related proceedings known to Appellants, Appellants' legal representative or Assignee which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.